National Institute of
Standards and Technology
U.S. Department of Commerce

General Electric
Corporate Research & Development

Virginie Galtier
Kevin Mills
Yannick Carlinet

Stephen Bush
Amit Kulkarni

Predicting and Controlling Resource Usage in a Heterogeneous Active Network

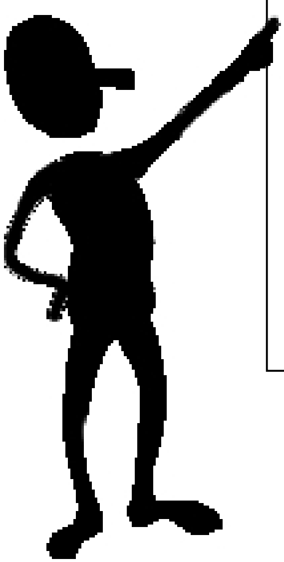
w3.antd.nist.gov/active-nets

Third Annual International Workshop on
ACTIVE MIDDLEWARE SERVICES

AMS

San Francisco
August 6, 2001

Outline of Presentation



❖ Motivations

❖ NIST solution to predict CPU requirements of an active packet on any node:

- Models in brief
- Prediction accuracy

❖ Application of NIST model to improve CPU-resource control in nodes

❖ Introduction to GE Active Virtual Network Management Prediction (AVNMP) a network load prediction system

❖ Enhancement of AVNMP by introduction of NIST models

❖ Future work

Motivations

Growing Population of Mobile Programs on Heterogeneous Platforms

SCRIPTING ENGINES & LANGUAGES

Using WinBatch

tcl tk

Python

vbscript
jscript

APPLETS & SERVLETS

JAVA

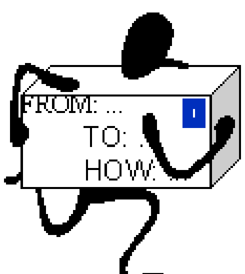
C#

dlls, dlls, and more dlls

MOBILE AGENTS

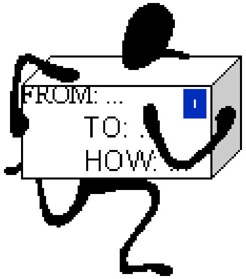
AgentSpace

D'Agents Dartmouth



Active Networks

Active Networks Overview



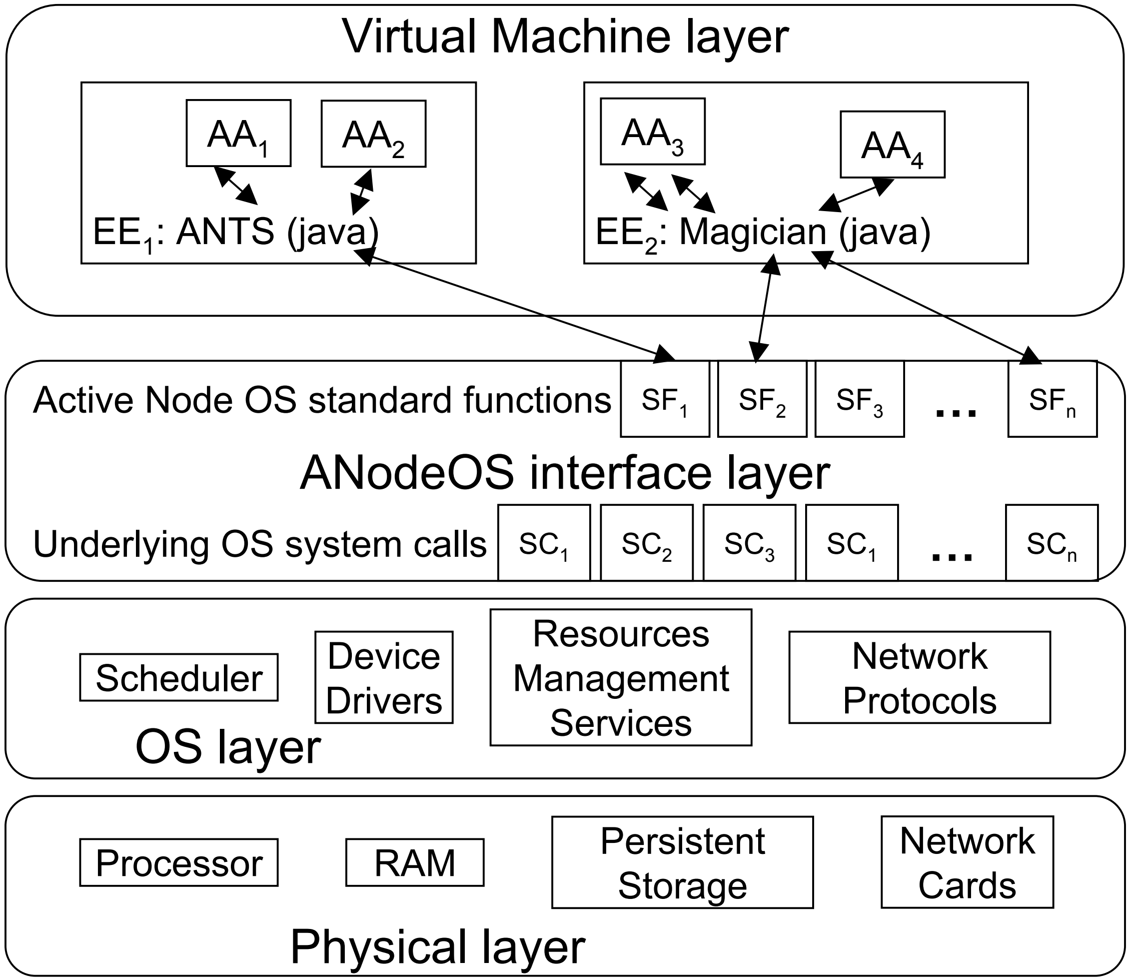
Principle: Active packets carry not only data but also the code to process them which is executed at active nodes.

Example: An application that sends MPEG packets can specify an intelligent dropping algorithm to be applied at intermediate nodes if congestion is detected.

Advantage: Fast and easy deployment of customized network services.

Motivations

Sources of Variability in Active Packet Execution Time



Motivations

Threats and Needs

Without a means to express and predict CPU cycles needed to execute an active packet:

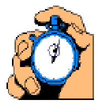
- Packets can consume excessive CPU time on a node or a set of nodes, causing denial of services to other packets
- A node can’t schedule its CPU resources to meet a packet’s performance requirements or other QoS requirements
- An active application can’t discover a route meeting its performance requirements
- Usage-based pricing simulations are impossible

NIST Model at a Glimpse

Monitor System Calls to NodeOS

AA$_4$

EE$_2$: Magician (java)



SF$_1$ | SF$_2$ | SF$_3$ | … | SF$_n$

ANodeOS interface layer

OS layer

Physical layer

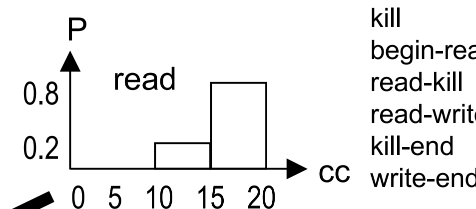
Generate AA Execution Trace

begin-4cc-read(20cc)-18cc-write(56cc)-5cc-end
begin-2cc-read(21cc)-18cc-kill(6cc)-8cc-end
begin-2cc-read(15cc)-8cc-kill(5cc)-9cc-end
begin-5cc-read(20cc)-18cc-write(53cc)-5cc-end
begin-2cc-read(18cc)-17cc-kill(20cc)-8cc-end

Generate AA Model on node X

Scenario A:
  sequence="read-write"
  probability=2/5
Scenario B:
  sequence="read-kill"
  probability=3/5

Distributions of CPU time in and between system calls:



Node Model

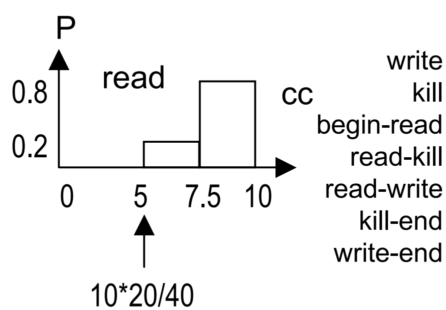
Model of Node X:
  read: 40 cc
  kill: 18 cc
  EE: 13 cc

Model of Node Y:
  read: 20 cc
  kill: 45 cc
  EE: 9 cc

Scaling AA Model

AA Model on node Y

Scenario A ="read-write", P=2/5
Scenario B ="read-kill", P=3/5



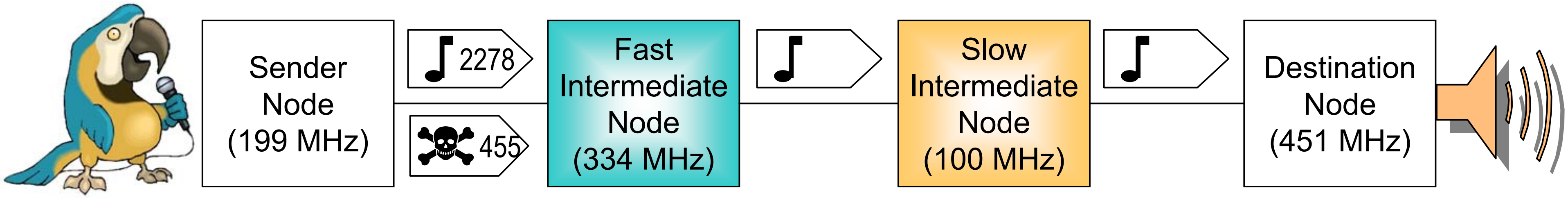
Multiple Monte Carlo Simulations

AA CPU requirements prediction on node Y

NIST Model Prediction Accuracy

EE	AA	Node X	Node Y	Predictions after scaling with speed ratio: Error on mean prediction	Predictions after scaling with speed ratio: Error on high percentiles prediction	Predictions with NIST model: Error on mean prediction	Predictions with NIST model: Error on high percentiles prediction
ANTS	Ping	machine A	machine B	94	110	0.42	8
		machine D	machine C	31	19	-2	8
		machine E	machine C	23	29	-7	7
	Multicast	machine B	machine E	22	20	-2	12
		machine C	machine D	-11	11	-2	10
		machine A	machine C	226	209	5	9
Magician	SmartPing	machine E	machine C	34	30	-5	9
		machine B	machine C	121	103	-7	14
		machine A	machine D	287	281	-9	10
	SmartRoute	machine E	machine D	14	10	-2	24
		machine D	machine C	15	21	-5	9
		machine C	machine A	-81	81	-3	10

Improved CPU Usage Control



Control = Kill packets which execute above 99th percentile of active audio packet execution time

Real:

8.29 ms = 1,650,084 cc

4.76 ms = 1,589,382 cc

23.99 ms = 2,398,702 cc

Experiment #1: predictions based on execution time on sender and processor speed ratio

8.29 ms = 2,769,487 cc

8.29 ms = 829,187 cc

Average execution time per packet:
(2278*M+455*8.29)/(2778+455)

2186 good packets are killed

Experiment #2: predictions obtained with NIST model

4.76 ms

Average execution time per packet:
(2278*M+455*4.76)/(2778+455)
Expected Improvement: 0.59 ms saved per packet
Experimental Result: 0.63 ms saved per packet!

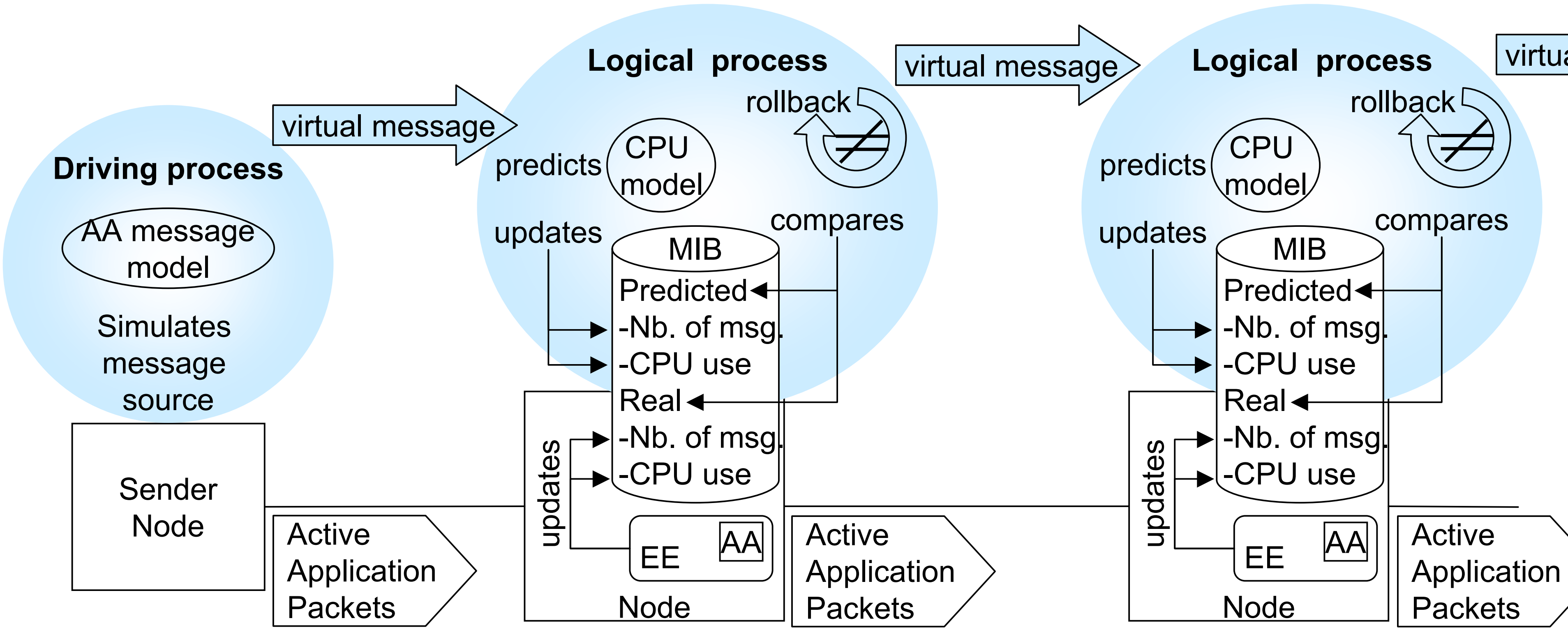
23.99 ms

Only 19 good packets are killed
Improvement = 2167 packets saved!

Improved Network Load Prediction

AVNMP in Brief

Overlay network simulates application traffic ahead in virtual time.



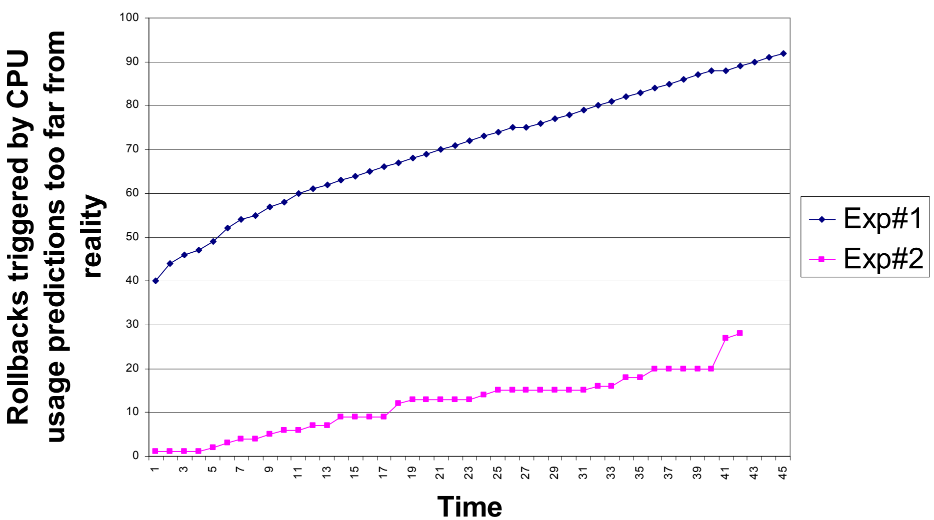
Experiment #1: CPU predictions based on average load on sender node and processor speed ratio
Experiment #2: CPU predictions obtained with NIST model
For both experiments: tolerance before rollback = 10 %.

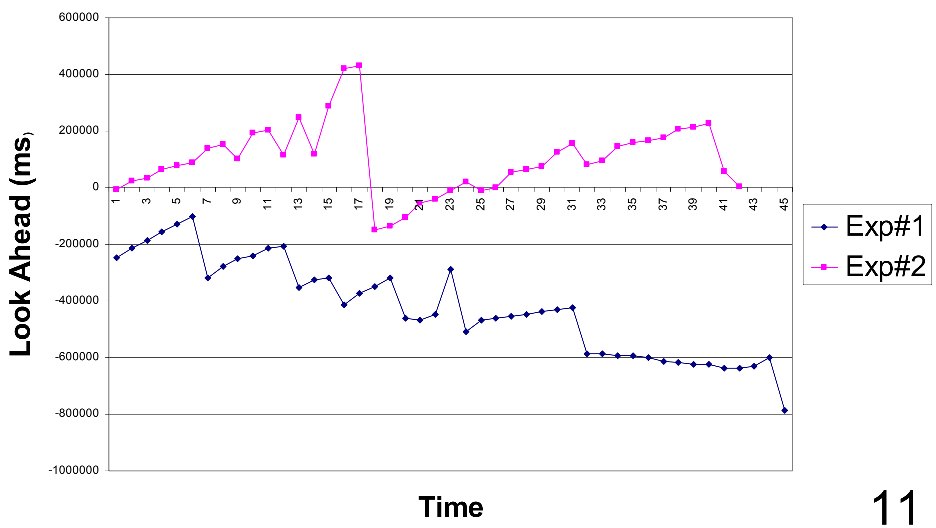
Improved Network Load Prediction

Experimental Results

	Exp#1: sender values scaled with processor speed ratio			Exp#2: CPU prediction with NIST model		
	first intermediate node	second intermediate node	destination node	first intermediate node	second intermediate node	destination node
maximum look ahead (seconds)	-101	-20	54	432	102	313
Rollbacks	92	42	12	28	0	0

AVNMP improvement on the first intermediate node:





Future Work

Improve NIST models

- trace-based model has limitations that could be overcome with models that learn or with models that consider node-dependent conditions
- investigate prediction based on competition
- investigate alternate models: white-box model currently underway
- characterize error bounds

Improve AVNMP performance

http://w3.antd.nist.gov/active-nets